JS - OAuth 2.0 with Google APIs - Overview

- Dr Nick Hayward

A brief overview of using OAuth 2.0 with Google APIs.

Contents

Intro

This outline is based on the Google Identity Platform documentation for OAuth 2.0 access to Google APIs.

Basic outline

There's a defined pattern to OAuth 2.0 based access to Google APIs. As an overview, there are primarily 4 main steps we may consider.

1. Credentials

Obtain OAuth 2.0 credentials from the Google API console. This provides a

- client ID
- client secret

These credentials are known to the developer's app and Google. The required credentials will vary from app to app, depending upon type. A JavaScript application, for example, will not require a *client secret*.

2. Obtain an access token

An app needs to obtain an access token from the Google Authorisation server before it can access private account data.

A single token may grant access to various APIs and different options. These options are dependent upon the granted *scope*, a variable parameter used to control the set of resources and operations that a token may access upon authorisation.

For a JS app, an app might request a token using a simple browser redirect to Google. For user content, authentication will be required using such a browser redirect. If the user authenticates correctly, Google will grant the app an access token. Otherwise, an error will be returned for the authentication failure.

3. Send access token to API

After successful receipt of an access token, an app can send this token to a Google API using a HTTP authorisation header.

Such access tokens are only useful and valid for the operations and resources specified in the *scope* of the token request. Access will often restricted to the specifed API.

4. Refresh access token

An access token has a limited lifetime. Once this lifetime has expired, it is possible to obtain a required *refresh* token.

This refresh token allows an app to obtain new access tokens.

JS usage

JavaScript apps need to work with Google's policies to enable authentication and authorisation for required APIs. For example, Google defines two levels of access for their APIs,

- simple
- authorised